Exploiting Future Word Contexts in Neural Network Language Models for Speech Recognition

X. Chen, X. Liu, Y. Wang, A. Ragni, J.H.M. Wong, M.J.F. Gales

***Abstract*—Language modelling is a crucial component in a wide range of applications including speech recognition. Language models (LMs) are usually constructed by splitting a sentence into words and computing the probability of a word based on its word history. This sentence probability calculation, making use of conditional probability distributions, assumes that there is little impact from approximations used in the LMs including: the word history representations; and approaches to handle finite training data. This motivates examining models that make use of additional information from the sentence. In this work future word information, in addition to the history, is used to predict the probability of the current word. For recurrent neural network LMs (RNNLMs) this information can be encapsulated in a bi-directional model. However, if used directly this form of model is computationally expensive when training on large quantities of data, and can be problematic when used with word lattices. This paper proposes a novel neural network language model structure, the succeeding-word RNNLM, su-RNNLM, to address these issues. Instead of using a recurrent unit to capture the complete future word contexts, a feed-forward unit is used to model a fixed finite number of succeeding words. This is more efficient in training than bi-directional models and can be applied to lattice rescoring. The generated lattices can be used for downstream applications, such as confusion network decoding and keyword search. Experimental results on speech recognition and keyword spotting tasks illustrate the empirical usefulness of future word information, and the flexibility of the proposed model to represent this information.**

***Index Terms*—Recurrent neural network, language model, succeeding words, speech recognition, keyword search**

I. Introduction

Language models (LMs) are crucial components in many applications, such as speech recognition and machine translation. The purpose of these language models is to compute the probability of any given sentence $\mathcal{W} = (w_1, w_2, ..., w_L)$. Language models are normally based on calculating the probability of current word based on its history words, these will be referred to as unidirectional LMs in this paper. This form of LM can be written as

$$P(\mathcal{W}) = P(w_0, w_1, w_2, ..., w_L) = \prod_{t=1}^{L} P(w_t|w_0^{t-1}) \quad (1)$$

X. Chen is currently with Microsoft, this work was done during he was in University of Cambridge. Y. Wang, A. Ragni, J.H.M. Wong and M.J.F. Gales are with the Department of Engineering, University of Cambridge UK (e-mail: xc257@eng.cam.ac.uk, yw396@eng.cam.ac.uk, jhmw2@cam.ac.uk, ar527@eng.cam.ac.uk, mjfg@eng.cam.ac.uk). X. Liu is with Chinese University of Hong Kong (e-mail: xyliu@se.cuhk.edu.hk). Software for the techniques described in this paper is available via the open-source toolkit CUED-RNNLM [3] in version 1.1: http://mi.eng.cam.ac.uk/projects/cued-rnnlm.

This requires the prediction of the probability of word w_t given its previous history $w_0^{t-1} = w_0, w_1, ..., w_{t-1}$. Two key research issues for LMs are: how to model long range dependencies; and how to handle data sparsity issues when modelling long-span contexts. n-gram LMs [4] and neural network based language models (NNLMs) [5], [6] are two widely used forms of language models. In n-gram LMs, a Markov assumption is made such that only the most recent $n-1$ words are used to represent the complete history. Thus

$$P(w_t|w_0^{t-1}) \approx P(w_t|w_{t-n+1}^{t-1}) \quad (2)$$

This form of truncated history context is also used in feed-forward NNLMs [5]. In contrast, recurrent neural network LMs (RNNLMs) model the complete history using a continuous, compact, vector space representation $\mathbf{h}_{t-1}$. The RNNLM predicted word probability is given by

$$P(w_t|w_0^{t-1}) \approx P(w_t|\mathbf{h}_{t-1}) \quad (3)$$

Most research on language modelling has focused on uni-directional LMs that only consider the word history. Future contexts may contain additional, useful, information for predicting the current word. The word probability was computed using $P(w_t|w_0^{t-1}, w_{t+1}^N)$ and the LMs built using this form of probability may contains complementary with traditional uni-LMs. There has been increasing research interest within the speech and language processing community in incorporating future word contexts to improve neural network language model performance [8], [9], [7]. For example, succeeding words were incorporated into RNNLMs within a maximum entropy framework in [8]. Forward and backward RNNLMs were separately constructed before being combined using a log-linear interpolation in [7]. [9] investigated the use of bidirectional RNNLMs (bi-RNNLMs) for speech recognition. On a broadcast news transcription task, the authors reported small improvements using future contexts in sigmoid activation based bi-directional RNNLMs, while no performance improvement was obtained using long short-term memory (LSTM) based bi-directional RNNLMs. To date, only limited and inconsistent gains in speech recognition performance have been reported with bi-RNNLMs over uni-directional RNNLMs (uni-RNNLMs).

In this work, bi-RNNLMs are first constructed and investigated for speech recognition. By applying a simple smoothing method and a two-stage interpolation between n-gram LMs, uni-RNNLMs and bi-RNNLMs can produce consistent, and significant, performance improvements over uni-RNNLMs on a range of speech recognition tasks. Though they can yield

performance improvements, bi-RNNLMs pose several challenges for both model training and inference as they require the complete previous and future word context information to be taken into account. First, it is difficult to parallelise training efficiently. Second, lattice rescoring is complicated for these LMs as future context also needs to be incorporated when determining the suitable merging and splitting of lattice paths. A range of efficient lattice rescoring techniques have previously been developed for uni-RNNLMs [11], [30], but these can not be used for bi-RNNLMs. Hence, N-best rescoring is used for these models [8], [9], [1]. In order to address these issues, a novel model structure, the succeeding word RNNLM (su-RNNLM) [2], is evaluated in this paper. Instead of using recurrent units to capture the complete future word context as in bi-RNNLMs, feed-forward units are used to model a small, fixed-length number of succeeding words. The existing efficient training [14] and lattice rescoring [11] algorithms developed for uni-RNNLMs can be extended to the proposed su-RNNLMs. This allows compact lattices to be generated with su-RNNLMs for many downstream applications. This paper is an extended version of previous conference articles [1], [2] that together describe the underlying theory of the models used. A more detailed discussion of the su-RNNLM and associated training is given here, along with updated results for the speech recognition tasks. In addition results on applying these advanced language modelling techniques to a keyword spotting task are presented.

The rest of this paper is organized as follows. Section II gives a brief review of standard unidirectional RNNLMs. Section III describes the structure of bidirectional RNNLMs (bi-RNNLMs). The proposed model with succeeding words (su-RNNLMs) is introduced in Section IV, followed by a description of the lattice rescoring algorithm in Section VII. Several practical issues for the use of bi-RNNLMs (and su-RNNLMs) are discussed in Section V. Experimental results are presented in Section VIII and conclusions are drawn in Section IX.

II. Unidirectional RNNLMs

In standard recurrent neural network language models (RNNLMs) [6], the history $w_0^{t-1} = w_0, w_1, w_2, ..., w_{t-1}$ of word w_t is represented using the 1-of-K encoding of the previous word w_{t-1} and a continuous vector $\mathbf{h}_{t-2}$, a compact representation of the remaining context w_0^{t-2}. Figure 1 shows an example of this unidirectional RNNLM (uni-RNNLM). The input consists of the most recent word w_{t-1}, which is projected into a low-dimensional, continuous, space via a linear projection layer. A recurrent, hidden, layer is positioned after this projection layer. There are many options for the recurrent unit used in the recurrent layer, e.g. standard sigmoid activations [6], or more complicated forms such as gated recurrent unit (GRU) [17] and long short-term memory (LSTM) units [18]. A continuous vector $\mathbf{h}_{t-1}$ representing the complete history information w_0^{t-1} can be obtained using $\mathbf{h}_{t-2}$ and previous word w_{t-1}. This vector is used as input of recurrent layer for the estimation of next word. The output layer with softmax function is then applied to calculate the probability $P(w_t|w_0^{t-1})$. An additional node is often added at the output layer to model the probability mass of out-of-shortlist (OOS) words to speed up softmax computation by limiting the vocabulary size [24], [19]. Similarly, an out-of-vocabulary (OOV) node can be added in the input layer to model OOV words. The probability of the word sequence $\mathcal{W} = w_0^L$ using these uni-RNNLMs is

$$P_u(w_0^L) \approx \prod_{t=1}^{L} P(w_t|w_{t-1}, \mathbf{h}_{t-2}) \approx \prod_{t=1}^{L} P(w_t|\mathbf{h}_{t-1}) \quad (4)$$

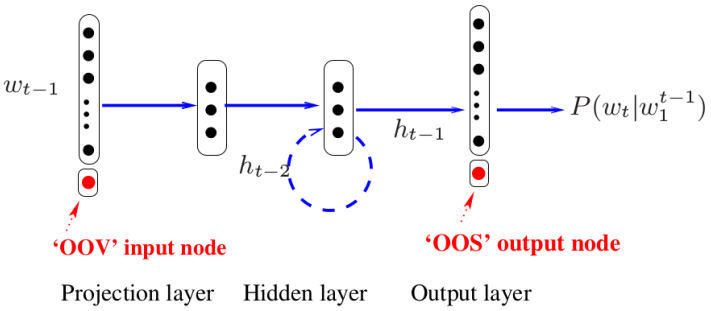

Fig. 1. *An example unidirectional RNNLM.*

The performance of a language model can be evaluated using Perplexity (PPL). Based on the definition from [20], the perplexity can be computed based on the average log probability at the sentence level using

$$\begin{aligned} \text{PPL} &= \exp\big(-\frac{1}{L}\log P_u(\mathcal{W})\big) \\ &= \exp\big(-\frac{1}{L}\log P_u(w_0^L)\big) \\ &= \exp\big(-\frac{1}{L}\sum_{t=1}^{L}\log P(w_t|w_0^{t-1})\big) \end{aligned} \quad (5)$$

Thus, for a unidirectional language model, the PPL can be calculated based on the average log probability of each word.

The training of uni-RNNLMs can be parallelised on Graphics Processing Units (GPUs) by using spliced sentence bunch (i.e. minibatch) mode [14], [16]. Multiple sentences are concatenated to form a long sequence and sets of these long sequences can then be aligned in parallel from left to right. In order to train sentence independent model, the sentence boundaries are marked and used to reset history vector. This data structure is very efficient for minibatch based training as they can be arranged to yield comparable total sequence lengths [14]. When using these forms of language models for applications such as speech recognition, N-best rescoring is the most straightforward way to apply uni-RNNLMs. Lattice rescoring can also become feasible by introducing some approximation [11] to simplify path merging and expansion in lattice. This will be described in more detail in Section VII.

III. Bi-directional RNNLMs

In the uni-RNNLMs, only history words are used to predict the probability of the current word. As previously discussed, though the sentence probability can be expressed in terms of

these conditional word-level probabilities, in practice this has ignored limitations in the conditional probabilities from the LM. One alternative option is to include future word context information. Bi-directional recurrent neural networks provide an option to incorporating this information. Figure 2 illustrates an example of bidirectional RNNLMs (bi-RNNLMs). Unlike uni-RNNLMs, both the history word context w_0^{t-1} and future word context w_{t+1}^L are used to estimate the probability of current word $P(w_t|w_0^{t-1}, w_{t+1}^L)$. Two recurrent units are used to capture the previous and future information respectively. In the same fashion as uni-RNNLMs, $\mathbf{h}_{t-1}$ is a compact continuous vector of the history information w_0^{t-1}. Additionally $\tilde{\mathbf{h}}_{t+1}$, another continuous vector, is added to encode the future information w_{t+1}^L. This future context vector is computed from the next word w_{t+1} and the previous future context vector $\tilde{\mathbf{h}}_{t+2}$ containing information of w_{t+2}^L. The history and future context vector $\mathbf{h}_{t-1}$ and $\tilde{\mathbf{h}}_{t+1}$ are concatenated and then fed into the output layer. The final probability is obtained using a softmax function. In order to reduce the number of parameter, the projection layer for the previous and future words are shared in this paper.

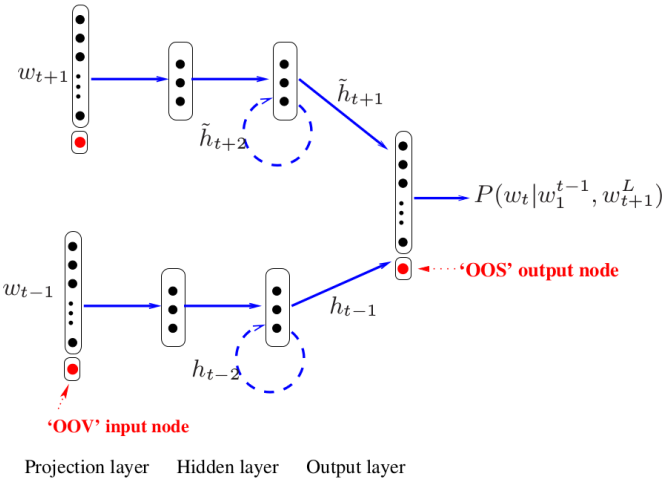

Fig. 2. *An example bidirectional RNNLM.*

Unlike uni-RNNLMs, the sentence probability of bi-RNNLMs can not be computed as a product of word probability when future information is taken into consideration. In the same fashion as the product of experts (PoE) framework, the probability of the word sequence $\mathcal{W} = w_0^L$ can be computed as,

$$P_b(w_0^L) = \frac{1}{Z_b}\hat{P}_b(\mathcal{W}) = \frac{1}{Z_b}\prod_{t=1}^{L} P(w_t|w_0^{t-1}, w_{t+1}^L) \quad (6)$$

$\hat{P}_b(\mathcal{W})$ is the unnormalized sentence probability computed using individual word probabilities from the bi-RNNLM. Z_b is a sentence-level normalization term to ensure the sentence probability is appropriately normalized, which is defined as,

$$Z_b = \sum_{\mathcal{W}\in\Theta} \hat{P}_b(\mathcal{W}) \quad (7)$$

where Θ is the set of all possible sentences. Unfortunately, this normalization term is impractical to calculate for most tasks.

In a similar form to Equation 5, the PPL of bi-RNNLMs can be calculated based on sentence probability as,

$$\begin{aligned} \texttt{PPL} &= \exp\big(-\frac{1}{L}\log P_b(w_0^L)\big) \\ &= \exp\big(-\frac{1}{L}\log\frac{1}{Z_b}\hat{P}_b(w_0^L)\big) \\ &= \exp\big(\frac{1}{L}\log(Z_b) - \frac{1}{L}\sum_{t=1}^{L}\log P(w_t|w_0^{t-1}, w_{t+1}^L)\big) \end{aligned} \quad (8)$$

In the last line of the above equation, the second term is the log word probability of bi-RNNLMs, which is similar to uni-RNNLMs. However, there is an additional term associated with Z_b, which is given in Equation 7, that is normally impractical to compute. As a result, it is usually not possible to compute a valid perplexity from bi-RNNLMs. Nevertheless, the average log probability of each word in bi-RNNLMs can be used to measure the accuracy of word prediction, which is referred as "pseudo" perplexity (PPL) in this paper.

$$\texttt{PPL}_{\texttt{pseudo}} = \exp\big(-\frac{1}{L}\sum_{t=1}^{L}\log P(w_t|w_0^{t-1}, w_{t+1}^L)\big) \quad (9)$$

This is a "pseudo" PPL because the normalized sentence probability $P_b(\mathcal{W})$ is discarded and the unnormalized sentence probability $\hat{P}_b(\mathcal{W})$ is used instead. Hence, the "pseudo" PPL of bi-RNNLMs is not comparable with the standard PPL of uni-RNNLMs. However, this "pseudo" PPL provides information on the average word probability from bi-RNNLMs since it is based on the individual word probabilities.

This expression illustrates the challenges of inference with bi-RNNLMs. It is difficult to combine bi-RNNLMs with other, often unidirectional, LMs such as n-gram LMs and RNNLMs since the sentence probability of bi-RNNLMs can not be computed. This will be discussed in Section V-A. Furthermore, N-best rescoring is normally used for speech recognition with these models[1]. Lattice rescoring is impractical for bi-RNNLMs as the word probability calculations require information from the complete sentence. However, lattices are very important in a range of downstream applications, including confidence score estimation [21], keyword search [22] and confusion network decoding [23].

Another potential drawback, limiting the use of bi-RNNLMs, is the difficulty of training the models. The complete previous and future context information is required to compute the word probability. It is computationally expensive to train bi-RNNLMs sentence by sentence, as well as difficult to parallelise the training for efficiency. The solution proposed in [9] is to concatenate all sentences in the training corpus together to form a single long sequence. This sequence can then be "chopped" into multiple sub-sequences with the same, data average, sentence length, enabling minibatch based training on GPUs. This allows bi-RNNLMs to be trained efficiently. However, in this fashion, the word is predicted using the "future" sentences (i.e. sentences after the current sentence) when the sub-sequence contains more than one sentence. This will introduce mismatch between training and real application if "future sentences" are not available. In this paper, bi-RNNLMs are trained in a more consistent fashion.

Unlike the training of uni-RNNLMs, where sentences are concatenated with spliced sentence bunch, for bi-RNNLMs training, multiple sentences are aligned from left to right to form minibatches. In order to handle issues caused by variable sentence length, `NULL` tokens were appended to the ends of sentences to ensure that the aligned sentences had the same length. These `NULL` tokens are not used for parameter update. Although resulting in slower training speed, it yielded more stable convergence and better performance than the approach in [9] in preliminary experiments.

IV. RNNLMs WITH SUCCEEDING WORDS

As discussed above, there are problems associated with bi-RNNLMs include the slow training speed and difficulties in lattice rescoring. In order to handle these issues, a novel structure, the su-RNNLM, is proposed to efficiently incorporate future context information. The model structure is shown in Figure 3. In the same fashion as bi-RNNLMs, the previous history w_0^{t-1} is modeled with recurrent units (e.g. LSTM, GRU). However, instead of modeling the complete future context information, w_{t+1}^L, using recurrent units, feed-forward units are used to capture a finite number, k, of succeeding words, w_{t+1}^{t+k}. When the succeeding words are beyond the sentence boundary, a vector of zeros is used as the word embedding vector, which is similar to the zero padding used in the feed-forward forward NNLMs [24]. These previous and future context information are merged in the output layer and a softmax function is applied to calculate the probability of the current word $P(w_t|w_0^{t-1}, w_{t+1}^{t+k})$. In this work the projection layers for previous word and future context are shared.

For each word, as the number of succeeding words is finite and fixed, its succeeding words can be organized as a n-gram future context and used for minibatch mode training as in feed-forward NNLMs [24]. Hence, su-RNNLMs can be trained efficiently in a similar fashion to uni-RNNLMs using the spliced sentence bunch mode [14].

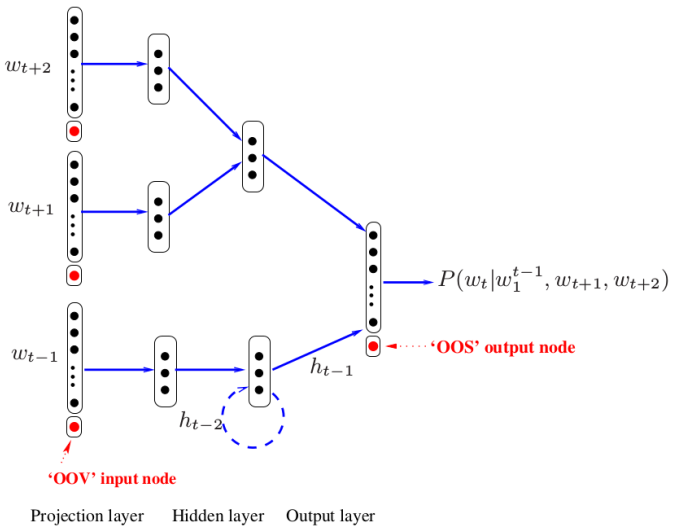

Fig. 3. *An example su-RNNLM with 2 succeeding words.*

Similar to the sentence probability of bi-RNNLMs as given in Equation 6, the probability of w_0^L can be computed as

$$P_s(w_0^L) = \frac{1}{Z_s}\prod_{t=1}^{L} P(w_t|w_0^{t-1}, w_{t+1}^{t+k}) \tag{10}$$

Again, the sentence level normalization term Z_s is difficult to compute and only "pseudo" PPL as defined in Equation 9 can be simply obtained.

V. USE OF BI/SU-RNNLMs FOR SPEECH RECOGNITION

As discussed previously, it is not possible to calculate the exact probability of the whole sentence, w_0^L, using both bi-RNNLMs and su-RNNLMs. This poses several theoretical and practical challenges when using these models for speech recognition. In this section, the interpolation of language models, and the possible sensitivity of systems to ASR errors, are discussed. As bi-RNNLMs and su-RNNLMs face similar issues when they are applied for speech recognition, only su-RNNLMs will be discussed here. The methods described in this section can be applied directly on bi-RNNLMs without modification.

A. Language Model Interpolation

n-gram LMs have been the dominant language models during the last several decades [25], [26]. Uni-RNNLMs were shown to present different and complementary modeling ability to n-gram LMs [27], [28]. Improved performance can be obtained by interpolating n-gram and uni-RNNLMs [27]. However, for su-RNNLMs, it is difficult to calculate the sentence level normalized probability thus traditional interpolation approaches involving su-RNNLMs are more complicated. Linear and log-linear interpolation are the two most popular approaches to combine multiple language models and will be discussed below.

1) ***Linear Interpolation****:* linear interpolation is widely used to combine multiple uni-LMs. Here considering the linear interpolation of n-gram and uni-RNN LMs,

$$P(w_t|w_0^{t-1}) = \lambda P_{rnn}(w_t|w_0^{t-1}) + (1-\lambda)P_{ng}(w_t|w_0^{t-1}) \tag{11}$$

where λ is the interpolation weight of RNNLM. The resulting interpolated probability is a valid probability mass function (PMF), which means that this interpolated LM is still a valid unidirectional language model. It can be used to calculate sentence probabilities using Equation 1.

However, applying linear interpolation to combine uni-LMs and su-RNNLMs would yield,

$$P(w_t|w_0^{t-1}, w_{t+1}^L) = \lambda P_{uni}(w_t|w_0^{t-1}) + \frac{1}{Z_{su}}(1-\lambda)P_{su}(w_t|w_0^{t-1}, w_{t+1}^L) \tag{12}$$

It is usually not practical to calculate the normalisation term Z_{su}, thus it is problematic to combine uni-LMs and bi-LMs using linear interpolation and obtain a valid PMF.

*2) **Log-linear Interpolation**:* an alternative approach to combine LMs is log-linear interpolation, which applies linear interpolation in the log domain [29]. For two uni-LMs this yields

$$P(w_t|w_0^{t-1}) = \frac{1}{Z(w_0^{t-1})} P_{rnn}(w_t|w_0^{t-1})^{\lambda} P_{ng}(w_t|w_0^{t-1})^{1-\lambda} \quad (13)$$

where $Z(w_0^{t-1})$ is a history-dependent normalisation term, which can be computed by summing over the vocabulary $\mathcal{V}$,

$$Z(w_0^{t-1}) = \sum_{w \in \mathcal{V}} P_{rnn}(w|w_0^{t-1})^{\lambda} P_{ng}(w|w_0^{t-1})^{1-\lambda} \quad (14)$$

where λ is the "weight" assigned to the uni-RNNLM [1].

Log-linear model combination with su-RNNLMs is again complicated by the need to compute normalisation terms when using future word information. To address this, models can be interpolated at the sequence, rather than word level. Thus considering the combination of a uni-LM and su-RNNLM

$$\begin{aligned} P(\mathcal{W}) &= \frac{1}{Z} P_{uni}(\mathcal{W})^{\lambda} P_{su}(\mathcal{W})^{1-\lambda} \\ &= \frac{1}{\bar{Z}} P_{uni}(\mathcal{W})^{\lambda} \hat{P}_{su}(\mathcal{W})^{1-\lambda} \end{aligned} \quad (15)$$

where $\bar{Z}$ is the sentence-level normalisation term and $\hat{P}_{su}(\mathcal{W})$ is the unnormalized sentence probability defined in Equation 6. Log probabilities are usually used in speech recognition, thus Equation 15 becomes,

$$\log P(\mathcal{W}) = C + \lambda \log P_{uni}(\mathcal{W}) + (1-\lambda) \log \hat{P}_{su}(\mathcal{W}) \quad (16)$$

where C is a constant and does not alter the rank ordering of hypotheses. Thus, sentence level log-linear interpolation of uni-LMs and su-LMs, without computing C, can be used for speech recognition. Though the performance of su-RNNLMs cannot be evaluated using perplexity, it is valid to evaluate su-RNNLMs in terms of ASR performance such as word error rate (WER). It is also worth noting that the sentence level log-linear interpolation is equivalent to the word level log-linear interpolation, as shown in Equation 15.

In this paper, we adopted a two-stage interpolation to combine n-gram, uni-RNN and su-RNN LMs. The n-gram and uni-RNN LMs are first combined using linear interpolation [2] and the resulted uni-LM is further log-linear interpolated with su-RNNLMs in sentence level.

In uni-LMs, the sentence probability was computed based on $P(w_t|w_0^{t-1})$. While in su-RNNLMs, the sentence probability is estimated based on $P(w_t|w_0^{t-1}, w_{t+1}^{N})$, although the sentence probability is difficult to compute. The combination of su-RNNLMs and uni-LMs can be operated on sentence level. As su-RNNLMs make use of future context for word prediction, it might contain complementary information when combining with traditional uni-LM estimated using Equation 1. In this paper, we used the same training data for construction of n-gram, uni-RNN and su-RNN LMs (bi-RNNLMs). Hence, we did not tune the interpolation weight and chose equal interpolation weight for simplicity. We also found the performances were not sensitive to the interpolation weight.

[1]The weights of the two models do not need to sum to one, but this simplifies the empirical tuning of the weights.

[2]linear interpolation and log-linear interpolation of n-gram and uni-RNN LMs gave similar performances.

B. Bi/Su-RNNLMs probability Smoothing

In [1], it was found that the word-level probability distributions from bi-RNNLMs are much sharper than those of uni-RNNLMs. This is easy to explain as the word-level predictions of bi-RNNLMs, which use both previous and future contexts, are much sharper than uni-RNNLMs where only history information is used. However, there is a potential drawback when using this sharp distribution for speech recognition. Bi-RNNLMs will be more sensitive to errors in the hypothesis from speech recognition systems, especially for tasks with high WERs. One approach to mitigate this effect is to smooth the well-trained bi-RNNLMs probabilities at inference time [1]. The smoothing algorithm can be written as,

$$P(w_i|w_0^{t-1}, w_{t+1}^{L}) = \frac{\exp(\alpha y_i)}{\sum_j^{\mathcal{V}} \exp(\alpha y_j)} \quad (17)$$

su-RNNLMs have the similar issue caused by the sharp probability distribution as bi-RNNLMs when the future information is used to predict current word. In this paper, probabilities from su-RNNLMs and bi-RNNLMs are smoothed using Equation 17. The smoothing factor α is chosen as 0.7 empirically. More details about the effect of smoothing procedure could be found in [1].

VI. NCE TRAINING OF SU-RNNLMS

Cross entropy (CE) is a widely used objective function for training neural network based language models. The probability of each word needs to be computed in CE training, which requires a normalization term to be computed over the whole vocabulary. This is computationally feasible for tasks with a small vocabulary, e.g. smaller than 20K words. However, the training is slow for tasks with a large vocabulary, e.g. larger than 50K words. In order to address this issue, noise contrastive estimation (NCE) has been applied to improve both training and evaluation efficiency [15], [14]. In NCE training, unnormalized probabilities are used during training, the normalization term is approximated by a constant that does not vary with the word context. During inference the unnormalized word probability can be used as the "constant" normalisation term will not alter the hypotheses ranking. Using NCE significant speedups can be achieved in both train and test time.

In this paper, NCE is applied to the training of su-RNNLMs on tasks with large vocabulary and large quantities of training data. As NCE training only affects the computation in the output layer, the NCE objective function can be directly applied to the su-RNNLM structure given in Figure 3. However, in initial experiments it was found that su-RNNLMs did not converge well using NCE training. Furthermore, the variance of the normalization term was found to be large, this impacts the accuracy of the inference process, this is illustrated in 5. Despite examining a range of hyper-parameter settings it was not possible to address these problems. A possible explanation

for this behaviour is that the context representations using recurrent neural networks and feed forward neural networks are very different. Thus, it can be challenging for the linear transform at the output layer to map these two representations into a "consistent" where a constant normalization term yields good language model parobabilities. In order to mitigate this issue, a modified su-RNNLM model structure is used, shown in Figure 4. An additional non-linear feed forward layer is added to combine the past and future contexts, this is the shared layer in Figure 4. In this work a sigmoid activation was used in this layer.

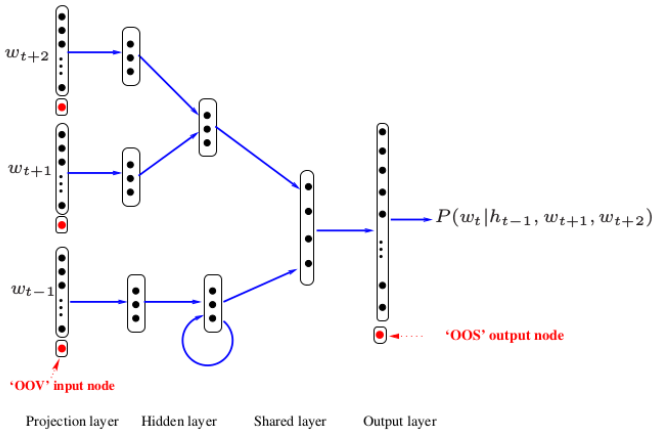

Fig. 4. *An example su-RNNLM with 2 succeeding words with an additional shared layer.*

Figure 5 shows the variance of the log-normalization term (over the observed contexts) for the original and modified su-RNNLMs architectures during NCE training on the AMI IHM (individual headset microphone) meeting corpus (see section VIII for details), evaluated on the validation data set. The green line shows the variance of the log normalization of the su-RNNLM without the shared feed forward layer (as shown in Figure 3). The red line corresponds to the su-RNNLM with a feed forward layer on top of the output layer (as shown in Figure 4). It can be seen that for the standard su-RNNLM without the shared feed forward layer, the variance of log normalization term increases significantly during training. In contrast, the variance is much smaller when using the modified su-RNNLM.

When using CE training the performance in terms of speech recognition was similar for the two su-RNNLMs architectures, Figures 3 and 4, indicating that the additional layer is not required for improved system performance with standard training. However, for NCE training, the modified su-RNNLM architecture, Figure 4 was found to be more stable and exhibit better convergence. su-RNNLMs with a shared layer were used in the experiments in Section VIII-B where the model was trained on large quantities of data, requiring NCE for efficiency. The standard su-RNNLM architecture was used in all other experiments.

VII. Lattice Rescoring

Lattice rescoring of feed-forward NNLMs with short context length (e.g. 3) is straightforward by expanding the lattice ac-

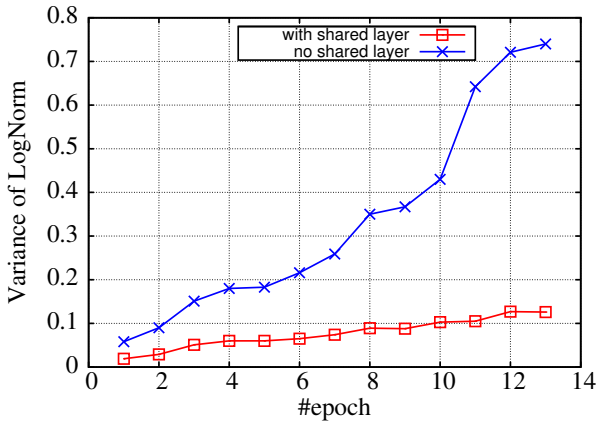

Fig. 5. *Variances of log normalization with and without the shared feed forward layer in su-RNNLMs on MGB3 data.*

cording to its n-gram history [24]. Additional approximations, discussed below, are required for efficient uni-RNNLMs lattice rescoring and resulting lattice generation [11], [12], [13], [30]. However, it is not simple to extend the uni-RNNLM approach to bi-RNNLMs as both the complete previous and future context information are required. As a result, N-best rescoring has previously been used for bi-RNNLMs. However, lattices are very useful in many applications. In contrast to bi-RNNLMs, su-RNNLMs only require a fixed number of succeeding words, instead of the complete future context information. Thus su-RNNLMs can be viewed as a combination of uni-RNNLMs for history and feed-forward NNLMs for future context. Hence, lattice rescoring is feasible for su-RNNLMs by extending the lattice rescoring algorithm of uni-RNNLMs to consider additional fixed length future contexts.

A. Lattice rescoring with uni-RNNLMs

In this paper, the n-gram approximation [11] approach is applied for uni-RNNLMs lattice rescoring. Here two paths are merged if the previous $n-1$ words of these two paths are identical. Figure 6 shows an example of partial lattice generated with a 2-gram LM. In order to apply uni-RNNLM lattice rescoring using a 3-gram approximation to this lattice, the grey shaded node in Figure 6 needs to be duplicated as word w_3 has two distinct 3-gram histories, which are (w_0, w_2) and (w_1, w_2) respectively. Figure 7 shows the expanded lattice after rescoring using a uni-RNNLM with 3-gram approximation. The history information from the best path is kept for the following RNNLM probability computation and the histories of all other paths are discarded. For example, the path (w_0, w_2, w_3) is kept and the alternative path (w_1, w_2, w_3) is discarded in arc w_4.

The implementation in this work differs slightly from that in [11]. There are two types of approximation involved in this uni-RNNLM lattice rescoring, which will be referred to as the merge and cache approximations. The merge approximation controls the merging of two paths described above. In [11] an additional approximation was used for implementation simplicity, the history of the first path reaching the node was kept and all other paths with the same n-gram history

were discarded irrespective of the associated scores. This introduces inaccuracies in the RNNLM probability calculation. To address this the path with the highest accumulated score is kept. Secondly, for fast probability lookup in lattice rescoring, n-gram probabilities can be cached using $n-1$ words as a key. A similar approach can be used with RNNLM probabilities, namely the cache approximation. In [11], RNNLM probabilities were cached based on the previous $n-1$ words, where n was constrained to be the same as the merge approximation. Thus a word probability obtained from the cache may be derived from a different complete history sharing the same $n-1$ previous words. The cache approximation here uses the complete history as the key for caching RNNLM probabilities. Both modifications yield small but consistent improvements over [11] on a range of tasks, at only a very small computational load increase.

B. Lattice rescoring with su-RNNLMs

Lattice rescoring with su-RNNLMs can be implemented by extending the n-gram approximation above to support future word context. In order to handle succeeding words correctly, paths will be merged only if the succeeding k words, as well as the previous $n-1$ words, are identical. Thus, the path expansion needs to be carried out in both directions.

Again considering a 3-gram rescoring history approximation, using the 2-gram lattice in Figure 6. In order to apply su-RNNLMs for lattice rescoring, the succeeding words also need to be taken into account. Figure 8 is the expanded lattice using a su-RNNLM with 1 succeeding word. The grey shaded nodes in Figure 7 need to be expanded further as they have distinct succeeding words. The blue shaded nodes in Figure 8 are the expanded node in the resulting lattice.

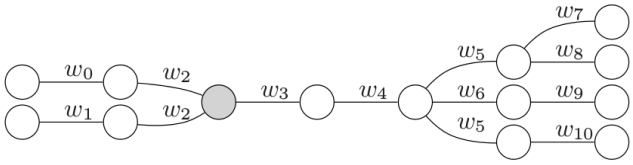

Fig. 6. *Lattice generated by 2-gram LM.*

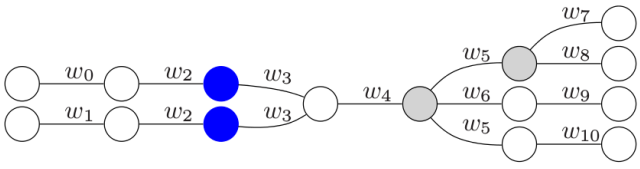

Fig. 7. *Lattice generated by uni-RNNLMs with 3-gram approximation.*

The computation cost of the lattice rescoring of su-RNNLMs with the n-gram history approximation and k succeeding words, is similar to an $(n+k)$-gram lattice expansion for uni-RNNLMs. For larger values of n and k, the resulting lattices can be very large. This can be mitigated by applying beam width pruning during the lattice expansion. In this paper, we only presented the WER results of su-RNNLMs lattice rescoring with 1 and 3 succeeding words.

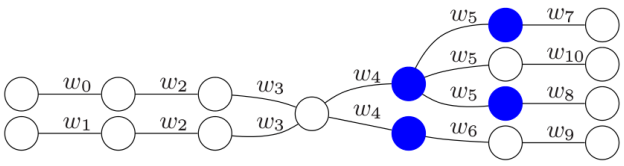

Fig. 8. *Lattice generated by su-RNNLMs with 3-gram approximation for history context and 1 succeeding word.*

VIII. Experiments

In this section, the performance of the su-RNNLM is evaluated for speech recognition and its impact on a down-stream processing task, keyword-spotting. The experiments are split into three sets. The first experiments examine the su-RNNLM for speech recognition in different configurations on a public data set, the AMI-IHM (individual headset microphone) meeting corpus. The second set of experiments illustrates the performance of su-RNNLMs when a large amount of training data is available, the MGB3 corpus. The final experiments investigate the performance of su-RNNLMs on a keyword search task using Babel-program data. Where results are stated as being significant, the matched pairs sentence-segment word error (MAPSSWE) based statistical significance test was used with a significance level of $p = 0.05$ and the tools provided by NIST[3].

A. Experiments on AMI-IHM data

The public AMI-IHM meeting corpus [36] was used to evaluated the performance of su-RNNLMs for speech recognition. The data configuration is the same as the Kaldi s5 recipe setup. A total of 78 hours of speech was used for acoustic model training. This consists of about 1M words of acoustic transcriptions. Eight meetings were excluded from the training set and used as the development and test sets. The TDNN-LSTM topology was used to build sequence-trained acoustic model [38] with the Kaldi toolkit [37].

The first part of the Fisher corpus, comprising 13M words, was used as additional language modeling training data. The decoding vocabulary consisted of 49k words. All LMs were trained on the combined (AMI+Fisher) 14M words. A 4-gram KN smoothed back-off LM without pruning was trained and used for lattice generation. GRUs were used as the recurrent unit for all unidirectional and bidirectional RNNLMs [4]. 512 hidden nodes were used in the hidden layer. The latest version of the CUED-RNNLM toolkit [3] was used to train the uni-RNNLMs, bi-RNNLMs and su-RNNLMs. The linear interpolation weight λ between the 4-gram LMs and uni-RNNLMs was set to 0.5. The log-linear interpolation weight, $(1-\lambda)$ in Equation 15, for bi-RNNLMs (or su-RNNLMs) was set to 0.3. The probabilities of bi-RNNLMs and su-RNNLMs were smoothed by a smoothing factor of 0.7. The 3-gram approximation was applied for the history merging of

[3]http://www1.icsi.berkeley.edu/Speech/docs/sctk-1.2/sc_stats.htm

[4]The GRU and LSTM gave comparable performances, while GRU LMs were faster for training and evaluation.

uni-RNNLMs and su-RNNLMs during lattice rescoring and generation [11].

Table I gives the training speed for the su-RNNLM, measured in words per second (w/s), and ("pseudo") PPLs with different amounts of future word context. When the number of succeeding words for the su-RNNLM is 0, it corresponds to the baseline uni-RNNLM. When the number of succeeding words is set to ∞, it is equivalent to a bi-RNNLM. It can be seen that su-RNNLMs can be trained at a similar speed to the uni-RNNLM when using limited future word contexts. The additional computational load of the su-RNNLMs mainly comes from the feed-forward unit for succeeding words as shown in Figure 3. The cost of this part is much less than that of other parts such as the output and GRU layers. Furthermore, the training of the su-RNNLMs is much faster than that of the bi-RNNLMs as the training su-RNNLMs is easy to parallelise efficiently. It is worth mentioning again that the PPLs of uni-RNNLMs cannot be compared directly with the "pseudo" PPLs of bi-RNNLMs and su-RNNLMs. But both PPLs and "pseudo" PPLs reflect the average log probability of each word. From Table I, with increasing number of succeeding words, the "pseudo" PPLs of the su-RNNLMs keeps decreasing. It is also worth noting that there are some minor differences between the estimation of sentence end </s> in bi-RNNLMs and su-RNNLMs. In su-RNNLMs, there is zero padding for the succeeding words it is beyond the sentence boundary. When all succeeding words are padded with 0, it strongly indicates that it is sentence end. However, for bi-RNNLMs, there is no padding to indicate the sentence end </s>. Therefore, pseudo-PPLs are not strictly comparable between su-RNNLMs and bi-RNNLMs.

TABLE I
TRAIN SPEED AND (PSEUDO) PERPLEXITY OF UNI-, BI-, AND SU-RNNLMS. 0 SUCCEEDING WORD IS FOR UNI-RNNLMS AND ∞ FOR BI-RNNLMS ON AMI-IHM DATA.

#succ words	0	1	3	7	∞
train speed(w/s)	4.5K	4.5K	3.9K	3.8K	0.8K
(pseudo) PPL	66.8	25.5	21.5	21.3	22.4

Table II gives the WER results of 100-best rescoring with different language models. As discussed in section V it is not possible to use linear interpolation for bi-RNNLMs (or su-RNNLMs). Log-linear interpolation was applied for the interpolation between these models and unidirectional LMs. The first block of Table II compares the WER results of uni-RNNLMs and bi/su-RNNLMs. The bi-RNNLM outperforms the uni-RNNLM by 0.2% absolute, the su-RNNLM yields the same performance as the uni-RNNLMs. The bi-RNNLM is statistically better than the uni-RNNLM and su-RNNLM on this task. The performance difference between the su-RNNLMs and uni-RNNLM is not statistically significant. In previous experiments on AMI-MDM (multiple distant microphones) and Babel "Dholuo" experiments [1], bi-RNNLMs were slightly worse than uni-RNNLMs. One possible explanation for the improvement of bi-RNNLM on the AMI-IHM task is that the reference segmentation is used here, and the WER is relative low compared to the AMI-MDM and babels tasks.

The second and third blocks in Table II show the WER results for the combination of three LMs, using a "reverse" uni-RNNLM or su-RNNLM to get future word context. It can be seen that increasing the number of succeeding words consistently reduces the WER. With 1 succeeding word, the word error rate was reduced by 0.2% absolutely. Su-RNNLMs with more than 2 succeeding words gave about 0.4% absolute WER improvement. These improvements using an interpolation of uni-RNNs su-RNNLMs (from 1 to 7 words) are statistically significant over the baseline uni-RNNLM. Additionally the gains from increasing of number of succeeding words are statistically significant when going from 0 to 1 to 2 succeeding words. The bi-RNNLM (shown in the bottom line of Table II) outperforms the su-RNNLMs by 0.1%, again statistically significant, as the recurrent units are more suitable to capture the complete future context information. Table II also presents the number of model parameters for different model structures in the third column. It can be seen that the bi-RNNLMs and su-RNNLMs have similar amounts of model parameters compared to uni-RNNLMs.

TABLE II
WERS OF UNI-, BI-, AND SU-RNNLMS WITH 100-BEST RESCORING. 0 SUCCEEDING WORD IS FOR UNI-RNNLMS AND ∞ FOR BI-RNNLMS ON AMI-IHM DATA.

LM	#succ words	#param (M)	dev	eval
4-gram		-	22.0	22.7
+uni-RNN	-	26.2	20.4	21.0
+bi-RNN	-	27.4	20.2	20.8
+su-RNN (3 su-words)	-	26.8	20.4	21.0
+uni-RNN+reverse-RNN	∞	52.4	20.0	20.6
+uni-RNN+su-RNN	0	26.2	20.4	21.0
	1	26.4	20.2	20.7
	2	26.6	20.0	20.6
	3	26.8	19.9	20.5
	4	27.0	19.9	20.5
	5	27.1	19.9	20.4
	6	27.3	19.9	20.4
	7	28.5	19.9	20.4
	∞	27.4	19.8	20.3

Another way to incorporate future word context for language modelling is to train reverse-order uni-RNNLMs from the end to the beginning of a sentence [7]. The two-stage interpolation was again used to combine the 4-gram, standard uni-RNN and reverse-direction uni-RNN LMs. WER results are shown in the third line of Table II. It can be seen that the reverse RNNLMs can yield a moderate WER improvement, but is slightly worse than su-RNNLMs and bi-RNNLMs. Note, this form of reverse RNNLM is also difficult to use with lattice rescoring for the same reason as bi-RNNLMs, as the complete future word context is required.

For the experiments in Table II, the su-RNNLMs were built using the configuration shown in Figure 3. An additional experiment was carried out to confirm whether su-RNNLMs with an additional shared feedforward layer (configuration in Figure 4) impacts performance when using the standard cross entropy training criterion. The experimental results show that these two su-RNNLM structures, both with 3 succeeding words, yield the same WER.

Table III shows the WERs of lattice rescoring using su-RNNLMs. The lattice rescoring algorithm described in Section VII was applied. As lattices are generated from this process, confusion network (CN) decoding was subsequently run on the genetated lattices, yielding additional performance improvements [5]. Su-RNNLMs with 1 and 3 succeeding words were used for lattice rescoring. From Table III, su-RNNLMs with 1 succeeding words gives 0.2% WER reduction and using 3 succeeding words gives about 0.5% WER reduction. These results are consistent with the 100-best rescoring result in Table II. Confusion network decoding can be applied on the rescored lattices and additional 0.2% WER performance improvements are obtained on the dev and eval test sets.

TABLE III
WERs OF UNI-RNNLMs AND SU-RNNLMs WITH LATTICE RESCORING ON AMI-IHM DATA

LM	#succ words	dev		eval	
		Vit	CN	Vit	CN
4-gram	-	22.0	21.8	22.7	22.5
+uni-RNN	-	20.4	20.2	20.9	20.7
+su-RNN	1	20.2	20.1	20.7	20.5
	3	19.9	19.7	20.3	20.1

B. Experiments on Multi-Genre Broadcast Data

This set of experiments investigates the use of su-RNNLMs on a larger corpus, the English Multi-Genre Broadcast (MGB3) challenge. Again TDNN-LSTM acoustic models were built using the Kaldi toolkit [37] using a total of 275 hours of data. More details of the acoustic model construction can be found in [40]. All language models were trained on 645M words, comprising 4M words of acoustic transcriptions and 640M words of subtitles. The vocabulary size was chosen to be 64K for building LMs. A 4-gram LM was used to generate lattices. Standard uni-RNNLM and su-RNNLMs were trained with the CUED-RNNLM toolkit [3]. Again GRUs were used as the recurrent unit. Given the increase in the quantity of training data, NCE training [14] was applied to efficiently train both the uni-RNNLM and su-RNNLM. RNNLMs were trained with 1024 hidden nodes and 1000 noise samples shared within each minibatch. It took about 3 days to train the su-RNNLMs on one K80 machine. As discussed in Section VI, an additional hidden layer was added in su-RNNLM, shown in Figure 4. The performance was evaluated on the dev17b test set distributed with the MGB3 challenge.

The experimental results are shown in Table IV. The WER improvements are consistent with the AMI-IHM system in the previous section. Standalone su-RNNLMs gave similar performance to uni-RNNLMs. The performance difference between the standalone su-RNNLM and uni-RNNLM is not statistically significant. 0.3% absolute improvement was obtained from Viterbi decoding, and 0.5% improvement from CN decoding, of by combining the uni-RNNLM and su-RNNLM. These improvements over the baseline uni-RNNLMs are statistically significant. This shows that the combination of uni-RNNLMs and su-RNNLMs performs well even on large amounts of training data.

[5]The N-best lists can be converted to lattices and CN decoding can then be applied, but it may require a large N-best list, such as 10K used in [11].

TABLE IV
WERs OF SU-RNNLM WITH 3 SUCCEEDING WORDS TRAINED ON 645M WORDS WITH LATTICE RESCORING ON MGB3 CHALLENGE

LM	WER	
	Vit	CN
4-gram	21.3	21.1
+uni-RNN	20.0	19.9
+su-RNN	20.1	20.0
+uni-RNN+su-RNN	19.7	19.4

C. Experiments on Keyword Search

Lattices are very useful for a range of downstream applications in spoken language processing. Thus developing language models that can act on, and generate, lattices is important. The application considered in this work is keyword spotting (KWS). The Swahili (IARPA-Babel202b-v1.0d) FLP from the Babel program was used for all experiments. In these experiments the performance of su-RNNLMs, which can be directly applied to lattices is compared to uni-RNNLMs. In [41] uni-RNNLMs were demonstrated to be effective for KWS. A total about 50 hours of transcribed conversational telephone speech data are provided to build the ASR and keyword search systems. All LMs were trained on the 400K acoustic model transcription with a vocabulary containing 24K words. The KWS performance was measured by maximum term weighted value (MTWV) in this paper, which is the TWV achieved at the optimal setting of the decision threshold in the DET curve [44].

Both ASR and KWS performance results are given in Table V. The first two lines show baseline systems using a 4-gram LM but with different acoustic models. The first line shows the CUED 2015 OpenKWS results. The second line presented the results of an improved acoustic model baseline with lattice-free MMI (LF-MMI) trained TDNN-LSTM [42], which yielded improved performances in both WER and MTWV. Since the In-Vocabulary (IV) performance for the keyword search system is most relevant for improved language modelling, indeed the language model weight for the out-of-vocabulary (OOV) search can be set to zero, only the performance of IV query terms is given. As shown in Table V, the standalone su-RNNLM with 1 succeeding words is worse than the uni-RNNLM and su-RNNLM with 3 succeeding words. The performance difference between the su-RNNLM with 3 succeeding words and the uni-RNNLM is not statistically significant. However, uni-RNNLMs achieved better KWS performances. The combination of uni-RNNLMs and su-RNNLMs consistently improves both ASR (statistically significant) and KWS performances, interpolating with the su-RNNLM with 3 succeeding words yields the best performance.

IX. CONCLUSIONS

Language models are an essential component of systems for many language processing tasks. Current state-of-the-art language models are usually based on recurrent neural networks

TABLE V
EXPERIMENTAL RESULTS ON KEYWORD SEARCH USING SU-RNNLMS ON SWAHILI BABEL CORPUS

System	WER	MTWV		
		IV	OOV	Total
OpenKWS 2015 CUED	44.7	0.5718	0.4264	0.5438
LF-MMI TDNN-LSTM	37.8	0.6333	0.3430	0.5783
+uni-RNN	37.0	0.6389	-	-
+su-RNN-1word	37.3	0.6343	-	-
+su-RNN-3word	37.0	0.6363	-	-
+uni-RNN+su-RNN-1word	36.8	0.6442	-	-
+uni-RNN+su-RNN-3word	36.6	0.6450	-	-

that use a compact history context representation to predict the next word. In this paper, models that enable the efficient use of future word context information for neural network language models have been described and evaluated. Initially Bi-directional recurrent neural network language models (bi-RNNLMs) were described. These provide a straightforward way to incorporate both past and future context information. However, they have some significant drawbacks: they are slow to train; and difficult to apply to lattice rescoring. This limits their potential applications. A novel model structure, the su-RNNLM, is proposed to address these issues. Instead of using a recurrent unit to capture the complete future information, a feed-forward unit is used to model a finite number of succeeding words. This structural change enables existing training and lattice rescoring algorithms for uni-RNNLMs to be extended for the proposed su-RNNLMs. Experimental results show that the combination of su-RNNLMs and uni-RNNLMs achieves significant performance improvements over standalone uni-RNNLMs. Further performance gains can be obtained with su-RNNLM by taking advantage of their ability to be applied in a lattice rescoring mode. To further demonstrate the advantages of this form of model, the combination of su-RNNLMs and uni-RNNLMs is shown to consistently outperform conventional model on a keyword spotting task using uni-RNNLM.

ACKNOWLEDGMENT

This research was funded under the ALTA Institute, University of Cambridge. Thanks to Cambridge English, University of Cambridge, for supporting this research. Xunying Liu is funded by MSRA grant no. 6904412 and CUHK grant no. 4055065. The authors would like to thank Ebru Arisoy, Abhinav Sethy and Bhuvana Ramabhadran from IBM for discussions and valuable suggestions on bidirectional RNNLMs.

REFERENCES

[1] Xie Chen, Anton Ragni, Xunying Liu, and Mark Gales, "Investigating bidirectional recurrent neural network language models for speech recognition.," in *Proc. ICSA INTERSPEECH*, 2017.

[2] Xie Chen, Xunying Liu, Anton Ragni, Yu Wang, and Mark Gales, "Future word contexts in neural network language models," ASRU, 2017.

[3] Xie Chen, Xunying Liu, Mark Gales, and Phil Woodland, "CUED-RNNLM an open-source toolkit for efficient training and evaluation of recurrent neural network language models," in *Proc. ICASSP*. IEEE, 2015.

[4] Stanley Chen and Joshua Goodman, "An empirical study of smoothing techniques for language modeling," *Computer Speech & Language*, vol. 13, no. 4, pp. 359–393, 1999.

[5] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin, "A neural probabilistic language model," Journal of Machine Learning Research, vol. 3, pp. 1137–1155, 2003.

[6] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model.," in *Proc. ISCA INTERSPEECH*, 2010.

[7] Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig, "Achieving human parity in conversational speech recognition," *arXiv preprint arXiv:1610.05256*, 2016.

[8] Yangyang Shi, Martha Larson, Pascal Wiggers, and Catholijn Jonker, "Exploiting the succeeding words in recurrent neural network language models.," in *Proc. ICSA INTERSPEECH*, 2013.

[9] Ebru Arisoy, Abhinav Sethy, Bhuvana Ramabhadran, and Stanley Chen, "Bidirectional recurrent neural network language models for automatic speech recognition," in *Proc. ICASSP*. IEEE, 2015, pp. 5421–5425.

[10] Bin Wang, Zhijian Ou, and Zhiqiang Tan, "Learning trans-dimensional random fields with applications to language modeling," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2017.

[11] Xunying Liu, Xie Chen, Yongqiang Wang, Mark Gales, and Phil Woodland, "Two efficient lattice rescoring methods using recurrent neural network language models," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 8, pp. 1438–1449, 2016.

[12] Michael Auli, Michel Galley, Chris Quirk and Geoffrey Zweig, "Joint language and translation modeling with recurrent neural networks," *Proc. EMNLP*, 2013.

[13] Martin Sundermeyer, Zoltan Tuske, Ralf Schluter and Hermann Ney, "Lattice decoding and rescoring with long-span neural network language models," *Proc. INTERSPEECH*, 2014

[14] Xie Chen, Xunying Liu, Yongqiang Wang, Mark Gales, and Phil Woodland, "Efficient training and evaluation of recurrent neural network language models for automatic speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2016.

[15] Michael Gutmann, and Hyvarinen Aapo "Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics," *Journal of Machine Learning Research*, 2012.

[16] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu, "Exploring the limits of language modeling," *arXiv preprint arXiv:1602.02410*, 2016.

[17] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[18] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] Junho Park, Xunying Liu, Mark Gales, and Phil Woodland, "Improved neural network based language modelling and adaptation," in *Proc. ISCA INTERSPEECH*, 2010.

[20] Frederick Jelinek, "The dawn of statistical asr and mt," *Computational Linguistics*, vol. 35, no. 4, pp. 483–494, 2009.

[21] Frank Wessel, Ralf Schluter, Klaus Macherey, and Hermann Ney, "Confidence measures for large vocabulary continuous speech recognition," *Speech and Audio Processing, IEEE Transactions on*, vol. 9, no. 3, pp. 288–298, 2001.

[22] Xie Chen, Anton Ragni, Jake Vasilakes, Xunying Liu, Kate Knill, and Mark Gales, "Recurrent neural network language models for keyword search," in *Proc. ICASSP*. IEEE, 2017, pp. 5775–5779.

[23] Lidia Mangu, Eric Brill, and Andreas Stolcke, "Finding consensus in speech recognition: word error minimization and other applications of confusion networks," *Computer Speech & Language*, vol. 14, no. 4, pp. 373–400, 2000.

[24] Holger Schwenk, "Continuous space language models," *Computer Speech & Language*, vol. 21, no. 3, pp. 492–518, 2007.

[25] Jerome R. Bellegarda, "Statistical language model adaptation: review and perspectives," *Speech Communication*, vol. 42, pp. 93–108, 2004.

[26] Roni Rosenfield, "Two decades of statistical language modeling: Where do we go from here?," *Proceedings of the IEEE*, 2000.

[27] Xie Chen, Xunying Liu, Gales Mark, and Phil Woodland, "Investigation of back-off based interpolation between recurrent neural network and n-gram language models.," in *ASRU, IEEE Workshop on*, 2015.

[28] Ilya Oparin, Martin Sundermeyer, Hermann Ney, and Jean-Luc Gauvain, "Performance analysis of neural networks in combination with n-gram language models," in *Proc. ICASSP*. IEEE, 2012, pp. 5005–5008.

[29] Alexander Gutkin, "Log-linear interpolation of language models," *Mémoire de DEA, University of Cambridge*, 2000.

[30] Martin Sundermeyer, Hermann Ney, and Ralf Schluter, "From feedforward to recurrent lstm neural networks for language modeling," *Audio,*

Speech, and Language Processing, IEEE/ACM Transactions on, vol. 23, no. 3, pp. 517–529, 2015.

[31] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Anton Ragni, Valtcho Valtchev, Phil Woodland, and Chao Zhang, "The HTK book (for HTK version 3.5)," *Cambridge University Engineering Department*, 2015.

[32] Brian Kingsbury, "Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling," in *Proc. ICASSP*. IEEE, 2009, pp. 3761–3764.

[33] Haipeng Wang, Anton Ragni, Mark Gales, Kate Knill, Phil Woodland, and Chao Zhang, "Joint decoding of tandem and hybrid systems for improved keyword spotting on low resource languages," in *Proc. ISCA INTERSPEECH*, 2015.

[34] Ebru Arisoy, Abhinav Sethy, Bhuvana Ramabhadran, and Stanley Chen, "Bidirectional recurrent neural network language models for automatic speech recognition," in *Proc. ICASSP*. IEEE, 2015, pp. 5421–5425.

[35] Xie Chen, Mark, Gales, Kate Knill, Catherine Breslin, Langzhou Chen, KK Chin and Vincent Wan "An initial investigation of long-term adaptation for meeting transcription," in *Proc. INTERSPEECH*. 2014.

[36] Jean Carletta et al., "The AMI meeting corpus: A pre-announcement," in *Machine learning for multimodal interaction*, pp. 28–39. Springer, 2006.

[37] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., "The Kaldi speech recognition toolkit," in *ASRU, IEEE Workshop on*, 2011.

[38] Karel Veselỳ, Arnab Ghoshal, Lukás Burget, and Daniel Povey, "Sequence-discriminative training of deep neural networks.," in *Proc. ICSA INTERSPEECH*, 2013.

[39] Mark Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Computer Speech & Language*, vol. 12, no. 2, pp. 75–98, 1998.

[40] Yu Wang, Xie Chen, Mark Gales, Anton Ragni, and Jeremy Wong, "Phonetic and graphemic systems for multi-genre broadcast transcription," in *Proc. ICASSP*. IEEE, 2018.

[41] Xie Chen, Anton Ragni, Jake Vasilakes, Xunying Liu, Kate Knill, and Mark Gales, "Recurrent neural network language models for keyword search," in *Proc. ICASSP*. IEEE, 2017.

[42] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, Sanjeev Khudanpur, "Purely Sequence-Trained Neural Networks for ASR Based on Lattice-Free MMI," in *Proc. Interspeech*. 2016.

[43] Mary Harper, "The BABEL program and low resource speech technology," in *Proc. ASRU*. 2013.

[44] "OpenKWS13 Keyword Search Evaluation Plan, "http://www.nist.gov/itl/iad/mig/upload/OpenKWS13-EvalPlan.pdf", 2013